Dav 3 - API Integration Report - Furniro

API Integration Process

Step 1: API Endpoint Setup

- API Endpoint: *https://677cca8c4496848554c7aa31.mockapi.io/products'
- Data Structure: The API provides product data, including fields like 'productName', 'category', 'price', 'inventory', 'colors', 's status', 'description', and 'image'.
- Library Used: Axios was used to fetch data from the API.

Step 2: Fetch Data from API

- The API call retrieves an array of product objects from the endpoint, ensuring proper data serialization.

Adjustments Made to Schemas

Product Schema Modifications

I. **Field Updates:**

- •colors': Added as an optional array of strings to accommodate multiple color options.
- 'images': Configured as a Sanity image field with hotspot enabled for better image cropping and scaling.

- category: added category for filter functionality

```
export const productSchema = {
    name: 'product',
    title: 'Product',
    type: 'document',
    fields: [
      {
        name: 'productName',
        title: 'Product Name',
        type: 'string',
      },
      {
        name: 'category',
        title: 'Category',
        type: 'string',
      },
      {
        name: 'price',
        title: 'Price',
        type: 'number',
      },
      {
        name: 'inventory',
        title: 'Inventory',
        type: 'number',
      },
      {
        name: 'colors',
        title: 'Colors',
        type: 'array',
        of: [{ type: 'string' }],
      },
      {
        name: 'status',
        title: 'Status',
        type: 'string',
      },
      {
        name: 'image',
        title: 'Image',
        type: 'image',
        options: {
          hotspot: true,
        },
      },
      {
        name: 'description',
        title: 'Description',
        type: 'text',
      },
    ],
}
```

1

1.

- Installed required dependencies: '@sanity/client', 'axios', 'dotenv'. - Created a '.env.local' file to securely store environment variables.

2. **Data Fetching:**
   - Retrieved product data from the API endpoint using Axios.
   - Parsed and logged the data to confirm its structure and integrity.

Migration Steps and Tools Used

Migration Steps

Environment Setup:

- Installed required dependencies: '@sanity/cliene, 'axios', 'dotenv'. - Created a '.env.locar file to securely store environment variables.

2. **Data Fetching:**

- Retrieved product data from the API endpoint using Axios.
- Parsed and logged the data to confirm its structure and integrity.

3. **Image Upload: **

- Downloaded images from the API's 'images field using Axios. - Uploaded images to Sanity's Asset Manager using the Sanity client. - Code to upload images:

```
const response = await axios.get(imageUr1, { responseType: arraybuffer•
const buffer = Buffer. from(response.data); const asset = await
client.assets.upload( image • , buffer, { filename: imageUr1. split( / ) .pop()
```

- Code snippet:

```
const sanityProduct = {
  _type: 'product',
  productName: product.productName,
  category: product.category,
  price: product.price,
  inventory: product.inventory,
  colors: product.colors || [],
  status: product.status,
  description: product.description,
  image: imageRef ? {
    _type: 'image',
    asset: {
      _type: 'reference',
      _ref: imageRef,
    },
  } : undefined,
};
await client.create(sanityProduct);
```

Tools Used

- **Sanity Client:** For interacting with the Sanity CMS.

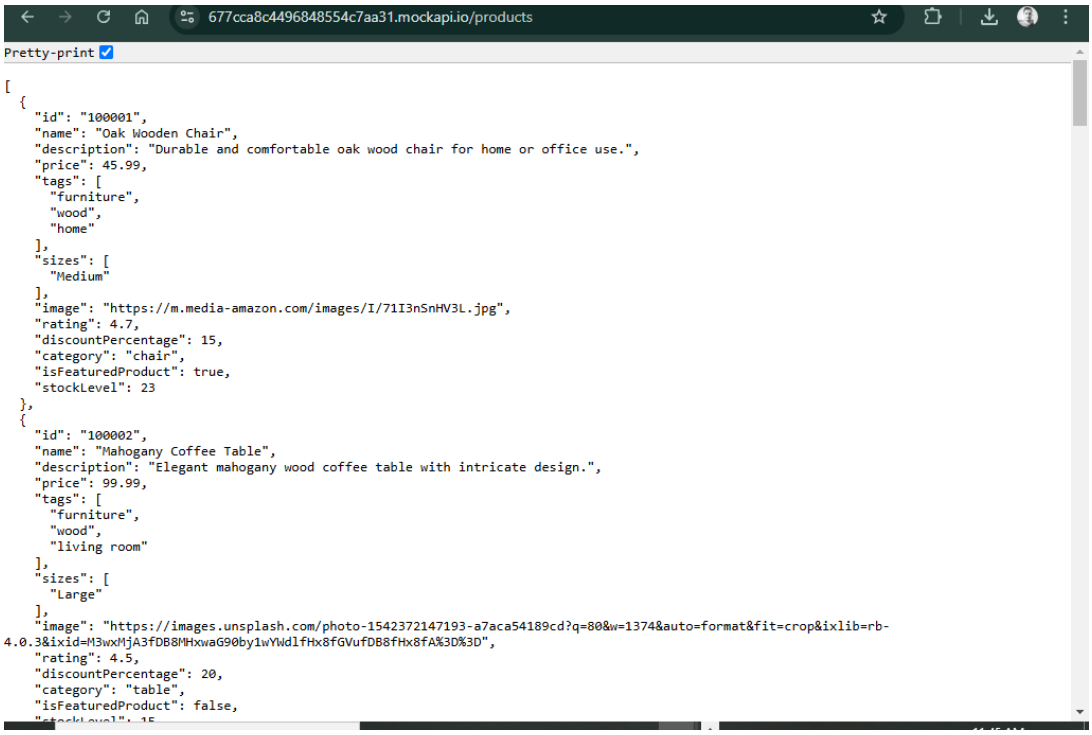
4. **Document Creation:**

- Created Sanity documents for each product by combining API data and uploaded image references.

- * *Axios:** For making HTTP requests to fetch API data and images.
  - **Dotenv:** To load environment variables from [s].env.local[S].

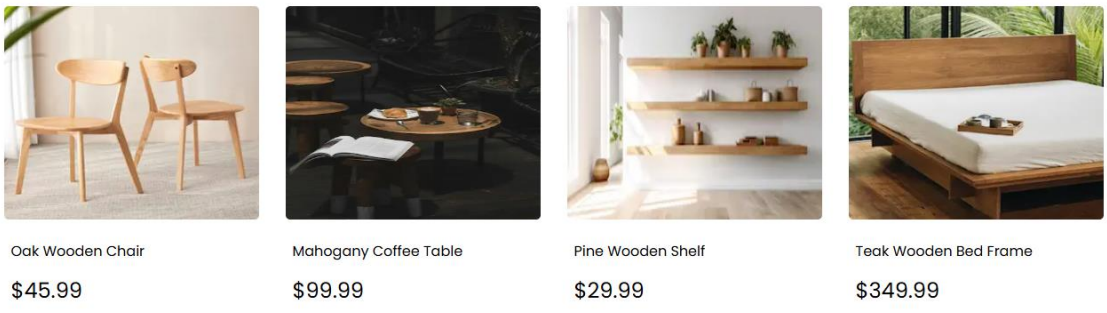
Screenshots

1. API Call Output

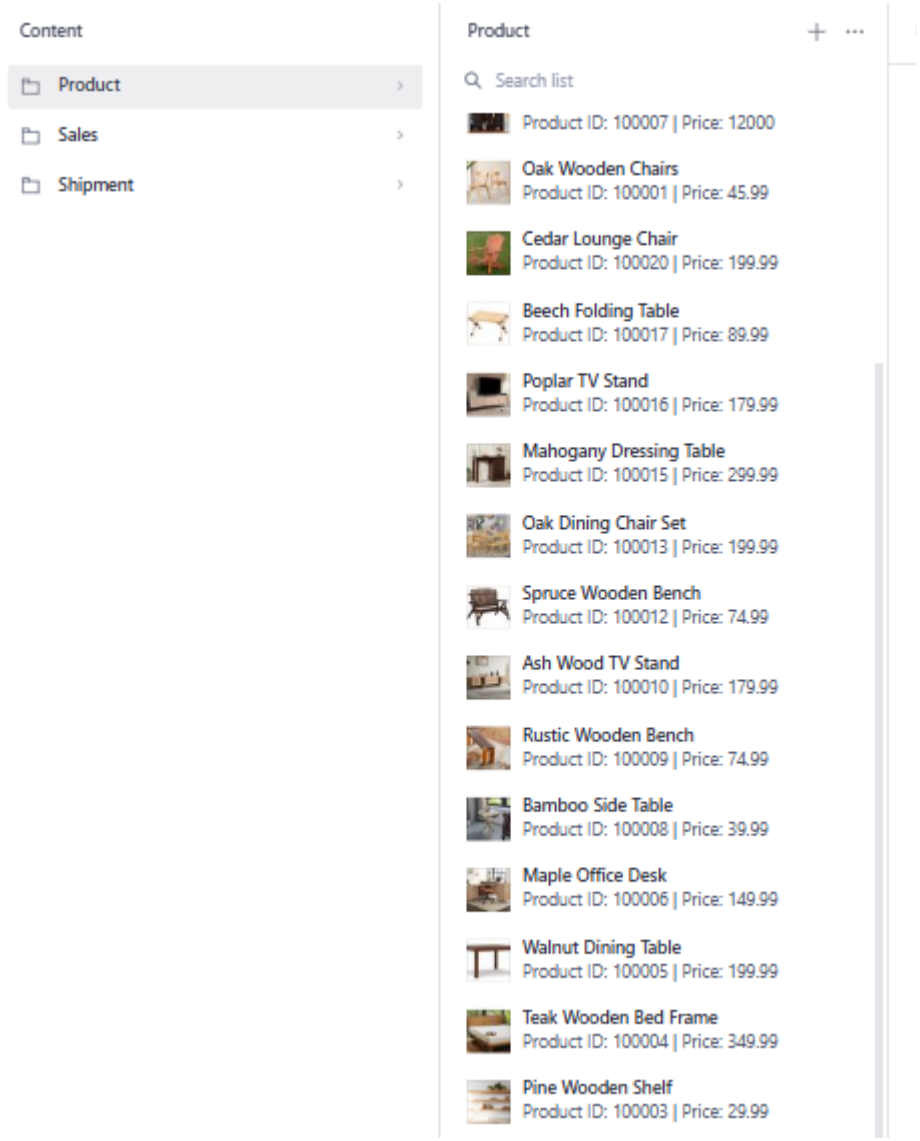


2.

3. Frontend Display



3. Populated Sanity CMS Fields



Code Snippets

API Integration

```
const response await
axios.gethttps://677cca8c4496848554c7aa31.mockapi.io/products); const products
= response.data.data;
```

Migration Script

```
async function uploadImageToSanity(imageUrl) {
  const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
  const buffer = Buffer.from(response.data);
  const asset = await client.assets.upload('image', buffer, {
    filename: imageUrl.split('/').pop()
  });
  return asset._id;
}

async function importData() {
  const response = await axios.get('https://template-03-api.vercel.app/api/products');
  const products = response.data.data;

  for (const product of products) {
    let imageRef = null;
    if (product.image) {
      imageRef = await uploadImageToSanity(product.image);
    }

    const sanityProduct = {
      _type: 'product',
      productName: product.productName,
      category: product.category,
      price: product.price,
      inventory: product.inventory,
      colors: product.colors || [],
      status: product.status,
      description: product.description,
      image: imageRef ? {
        _type: 'image',
        asset: {
          _type: 'reference',
          _ref: imageRef,
        },
      } : undefined,
    };

    await client.create(sanityProduct);
  }
}

importData();
```

Final Check List

Understan	Schema Validation	Data M ation	API Integration in	Submission Pr •ration
✓	✓	✓	✓	✓